Automation Of Supplier Data Validation

Software Documentation

Isha Thakur

05/03/2021

SUPPLIER DATA VALIDATION AUTOMATION TOOL

Edited By	Version	Date
Isha Thakur	1.0	05.03.2021

1. About

Our task is to automate the supplier data validation, the process of validating the supplier data for the purpose of data conversion is broadly divided into two different but sequential processes , namely Mapping and Validation.

Here About section is like an introduction to the project. For someone completely unaware of the work that needs to be done, it would be better if its bit more detailed, maybe in form of one liner's to address the questions that might come to mind. For example:

1. What is supplier data here? (Guessing: Is it some customer record or payment-invoice record?)
2. **What is the need of conversion of data? Please mention the reason in brief.**

Mapping: In this activity, we take the two input excel files which are from the source and target respectively. What does source and target refer to here?

To enable a data integrity check between the two files and to validate the files , the Mapping step is mandatory and very important.

In it , for certain business defined columns of the supplier data, a mapping file is required from the source records to the target records. The exact, partial and the mismatches are documented via the program and displayed in the Mapping file.

This allows an ease in manual overview to find the discrepancies and set them in a metric.**The key element to be kept in mind is that this source to target mapping can be either one-to-one or many-to-one but never one-to-many.**

1. When you are referring to the excel files please give a link to the file as well. I scrolled down the document to check if snaps/files are attached or not, I found snaps at the end of the document. Its inconvenient for user to read and scroll down and tally with the snaps. Please give a link here so that it can be matched side by side.
2. Okay, referring to the snap I am considering that source is **Zeal Shenyang Supplier Raw Data (Its initial ERP?)** and target is **Races Shenyang UAT Load Supplier Data (it's the final ERP?).** Please give the source and target details in brackets the first time you mention them.

3. As per the snaps attached, the result comes as True only when GSL9 value is same between 2 data.
    a. What is GSL9?
    b. Why is it that when GSL9 is same, you do not need to check other parameters?

Validation: The successive step to Mapping comes the Validation, in this the key (here GSL9 value) column of the Mapping , are used to retrieve the records from the source file and the target file and fill in the new validation file. We then compare the columns to identify the matches between the records that were mapped according to the key column.On obtaining this file, a summary report is generated by the program indicating the exact number of true, false and blank for a particular column and for all the records in the validation file.This summary report is crucial and handy in reaching out to other dependent teams to update/modify/delete the records that have issues.

1. Here you mentioned that GSL9 value column of the Mapping file is used to extract record from source and target. As per the snaps attached, there is Source GSL9, Target GSL9 and a GSL9 column which is not visible properly. Are all columns with GSL9 needed or is it only a specific column?
2. VAT column is not visible in sample test cases and Mapping file.

2. Purpose

The Supplier Data Validation Tool is used by the business to validate and ensure the data integrity during the process of conversion of data from one ERP to another ERP. The process is broadly divided into two major subprocesses, i.e generating a Mapping File and the Validation File. These are then forwarded to the next team for further data processing and data correction.

So keeping the above in mind, our program is also divided into two separate applications that perform the two subprocesses.

SubProcess	Functionality
Mapping Okay, so this is a very good page. Precise and well explained. Here I got to know that the source and target are 2 ERPs. Additionally, I understood that the supplier data is Vendor name, supplier name, number etc. Instead of the 2 pages before, you can start with this page. Here you put out things very well, more information in concise way as compared to the 2 pages above. In my opinion the lesser the pages in documentation the better. **Please put the link to Mapping file here.**	In this process, we take the source and the target ERP files as our input. Then we map certain columns of both the files to each other , to check if the particular source record exists in the target file or not. This is broadly done on the basis of values five column parameters: ❖ Map using GSL9 (Vendor Num + Supplier_site) ❖ Map using ScxGSL (SCxID + Supplier_Site) ❖ Map using Vendor Number Only ❖ Map using Supplier Name Only ❖ Map using Address Only Any remaining record should be marked as 'FALSE', and will need to be mapped manually later.
Validating Once again, in my opinion here you put out information very well. **Please give link to Validation file here so that user can refer while reading.** 1. Which GSL9 values? **Is it source and target GSL9 only to get the row index in source and target respectively?** 2. **Point 2, why will be there a multiple match? You mentioned above that the mapping can never be one to many so how can there be multiple match at target for a value in source?** 3. In the last point you mentioned that comparison is run between the corresponding source and target column in validation file. **Please mention the column names taken into consideration for comparison. Going by the snap you attached for the validation file, I see only GSL9 columns with heading as source and target. Are these 2 columns only compared?** Please bring clarity here so that user can refer and understand.	In this process, we take the source, target and the previously generated mapping file as our input. Then the following steps are followed: ❖ We use the GSL9 values of the mapping file , to find the corresponding row index in the source and target files. ❖ In an event of multiple matches, we further create a combination of certain columns, and the record matching the most on these values are selected. ❖ Once the row index of the record in mapping file is found in the source and the target file, it is used to populate the rest of the columns in the validation tracker. ❖ Then a comparison is run between the corresponding source and target column in the validation file.

3. Scope

The scope of this application is limited to the GE Renewable DT ERP Data Conversion team. This application was developed keeping in mind the above mentioned business requirements at the time of development. It has proven to work on Oracle ERP extracted files.
The application needs a buffering time of 5 seconds after exporting and before shutting the application.

4. Assumption

- The column names for the same field will be according to a set standard format in both the excel files.
- The program has been implemented keeping in mind the Oracle ERP format in consideration.
- Only excel files can be input and output.
- GSL9 values are previously present in the input excel files.
- Both the source and target files are for the same business and same location.
- The output is expected as of the business logic provided as part of requirements elicitation.

5. Input Dataset:

Our inputs for the mapping file generator will be two excel files, where one is from the source (ex. Zeal) and the other is of the target excel file (ex. Races) .

If the user didn't refer to the snaps at the end of the document, then this will be the first time he would know what is source and target file. Please mention in the beginning as well.

Similarly , for the Validation File Generator we will have the input as the Source excel file, the target excel file and also the previously generated mapping file from the mapping file generator.

Since you have defined this before in validation part under process, there is no need of the input dataset part in my perspective. You may remove it.

MAPPING FILE GENERATOR TOOL

Programming Language Used

Python

Libraries Imported

Imported Libraries
import datetime
from datetime import datetime
import numpy as np
import regex as re
from re import search
import string
import pandas as pd
import tkinter as tk
from tkinter import datetime
from datetime import datetime
import numpy as np
import pandas as pd
import tkinter as tk
from tkinter import filedialog, Label
from tkinter import messagebox

Programming Application Generation

The conversion from Python notebook to a python file was initially done. Then the python file was converted into .exe application using pyinstalle

Algorithm Steps

Step 1 : Import the Source File

```
def getExcel():
    """
    Gets the source excel from the user and replaces the nan values with ' '.
    """
```

Step 2 : Import the Target File

```
def getExcel1():
    """
    Gets the target excel from the user and replaces the nan values with ' '.
    """
```

Step 3 : Doing the Mapping

```
def cmp():
    """
    Does the actual mapping and stores the output in a global variable to be later
exported.
    Converts the imported files into pandas dataframe.
    Adds the' SCxGSL' and the 'concatenated address' column to the source and target
files                         (What is the concatenated address col? How is it found?)
   Has several nested functions as listed below:
```

```
def compare_BankEndDate(df, r):
    """
    :param df: The dataframe where the status of bank_end_dates need to be checked
    :param r: The no. of rows of the dataframe


   This function for df checks for the activeness  of the bank accounts.
   If any date in the   'BANK_END_DATE' column is of a date prior to the current date,
i.e now,
    then it marks it as FALSE. Any other date or a blank value is assigned as TRUE.

    """
```

```
def   drop_valid(df):
    """
    :param df: The processes dataframe where the status of bank_end_dates need to
be checked

    Drops the inactive bank records and the rows from the dataframe based on the
TRUE/FALSE values.

    """
.
```

```
def preprocess_str(df):
    """


    Args:
```

df: The specified column of the dataframe to preprocess them
""" """

Converts the certain columns into processed columns by removing any punctuation and converting all the text to lower form and eliminating any space

Please keep the variable name constant across functions. Don't use same variable name for multiple variables. Reason behind is this that code doesn't look clean, second it reduces readability of code and creates confusion. Here you considered 'df' as specific column, before you used it for entire data frame.

def map_Files(source, target, column, df):
"""

The driver code for mapping.
:param source: The first dataframe - source
:param target: The second dataframe - target
:param column: The column//will we consider all columns one by one? If so where will you iterate over the columns and when will we break out of it?
:param df: The mapping_file
Here you are using df for mapping file, so now the user thinks that ..okay, before df was dataframe, then it was used for specific column and now it is mapping file. This creates inconvenience.
We iterate over the length of the source file, and it stores the value obtained for a particular row index and the column in a variable named 'key'. (Length of src file means the number of rows right?)
We then call the function 'getTargetIndex' discussed below in the other cells.
This function call returns the corresponding races row index for the 'key', and stores it in the variable 'trgt_ind'.
If the trgt_ind is not equal to -1 , which means a match, we loop over our necessary and predefined list of columns, and extract the values from the source and target files to populate their respective columns in the mapping file.
If the trgt_ind is equal to -1, it means no match for that key was observed in the target file(What will be next step when its -1? Will the func be called again for next column?)
So unless and until that target value in the mapping file was already filled any any previous comparison , we assign it a blank value for the target column in the mapping file. The source column in the mapping file is filled by key.
Here you are iterating through rows of column at index 0 i.e. GSL9 column? Now if GSL9 value doesn't gets matched somewhere in the target column index 0, then it will be -1 at trgt-ind, correct? If its not -1 then for the same row in src, we iterate across columns and populate the mapping file columns, plus we iterate across the columns for row 'trgt_ind' in target file and populate cols in mapping file, correct? If its -1 we will move to column at index 1?
"""

def getTargetIndex(key, target, column, src_ind, df):
"""

Returns the corresponding best matched target index
:param key: The source file value for the particular column and the src_ind **
:param target: It is our second dataframe
:param column: The column header in consideration
:param src_ind: The current position or row index
:param df: The mapping_file
:return: The corresponding target row index for the key

Here we find the corresponding matched row indices in the target file for the key and store it in ind_values.
If the length of ind_values is equal to 1, then the 'MULTIPLE_MATCH?' column of the mapping file is set to 'FALSE', due to the absence of more than one match.
(Again, why will there be a multiple match if one-many map isn't possible)
Else If, the length of ind_values is greated to 1 , then the 'MULTIPLE_MATCH?' column of the mapping file is set to 'TRUE', due to the presence of more than one match.
To find the correct match we use fuzzy logic between the key "Payment_Term_Match" column (i.e the source) (PTM not visible in snap, what is PTM?) and all the "Payment_Term_Match"values of the cells from the indices in ind_values.(trgt vals?)
This is stored as a tuple , in the form of (the target row index, fuzz ratio).
All the tuples obtained for the length of ind_values is then stored in a list , which gets sorted according to the fuzz_ratio in the tuple, using the function call of Sort_Tuple().
We just then return the highest scoring target index value.

Else If the length of the ind_values is 0, then we return -1, indicating a not_match.

```
    """
```

```
def Sort_Tuple(tup):
    """
    Sorts a list of tuple, based on the second value in the tuple
    :param tup: A list of tuples
    :return: A sorted list of tuples
    """
```

```
  def check(z, r, m, df): what is purpose of check function?
    """


    :param z: The source file in mapping_file (you mean source column in map file?)
    :param r: The target file in mapping_file(you mean trgt column in map file?)

    :param m: The matching column in mapping_file
    :param df: The mapping_file
```

Once the mapping file is populated with the appropriate source and target values, we compare the source and target columns for the entire df.
If the source and target values are equal and they are not null, it is assigned as 'TRUE'.
Else If the source and target values are equal and they are null, it is assigned as 'NULL'.
Else it is assigned as 'FALSE'. (its difficult to visualise via text, please show some example here via snap..so that its easy to tally with text. When you are comparing the src and trgt col in mapping file, you might as well state as, 'here we are comparing column name this with column name this' if you do not wish to attach snap)

```
        """
```

```
    def checkConcat(x, y, df): Why this function? What is purpose?
        """
        It will make the 'x' value as 'FALSE' if 'y' is 'NULL' for the respective row
        @param x: Concatenated Column Match Column
Where is this concatenated col is coming from? Purpose of X and what is X is not clear.
        @param y: Part of Concatenated Match Column
        @param df: mapping_file
        """
```

```
    def result(df):
        """
        Fills the 'MATCHED' and "MATCHED_ON?" columns
        :param df: The mapping_file

We maintain two list of values, one is for the column 'MATCHED?' and the other one is for 'MATCHED_ON?' column in the mapping file.


If the record in Mapping File is matched on GSL9 , then it's TRUE.
Else if the record is matched on SCxGSL , the it's TRUE.
For anything else it is a PARTIAL MATCH.
If the record is matched only on Supplier_Site or not on any column, it is assigned 'FALSE'.



 """
```

```
    """
```

Step 4 : Export the Mapping File

```
def putExcel():
   """
  Saves the mapping_file to the specified location in the user system.
   """
```

When writing the code description, a good practice is:

1. **Making the flowchart(most imp):** So that user understands that which function relates to which other functions. Otherwise with multiple functions here, user loses track and gets confused. A diagram at the top helps prepare the user for the flow to come.
2. **Keeping the code clean**: Do not use same name for multiple variables. Example**: 'df '** in this case. Keep variable name as 'df_col' when you refer to a particular column in data frame, similarly keep mapping file name as 'mf '. When you use same name for multiple variables across different functions it will create confusion not only for the reader but also for you the next time you go through the code because the same variable name has different functionality at different point.

3. **Improving code readability**: Code should be in a way that a user who doesn't have any insight can understand the code easily by reading it. Plus the next time you go through the code, it will help you comprehend as well. For that, name the variables by its functionality. You used z and r for columns of file. Instead of that use something which user can read and understand then and there what the variable is for.

VALIDATION FILE GENERATOR TOOL

Programming Language Used

Python

Libraries Imported

Imported Libraries
import datetime
from datetime import datetime
import numpy as np
import pandas as pd
import tkinter as tk
from tkinter import datetime
import regex as re
from re import search
from datetime import datetime
import numpy as np
import pandas as pd
import string
import tkinter as tk
from tkinter import filedialog, Label
from tkinter import messagebox

Programming Application Generation

The conversion from Python notebook to a python file was initially done. Then the python file was converted into .exe application using pyinstaller.

Algorithm Steps

Step 1 : Import the Source File

```
def getSource_Excel():
   """
   Gets the source excel from the user and replaces the nan values with ‘ ‘.
   """
```

Step 2 : Import the Target File

```
def getTarget_Excel():
   """
   Gets the target excel from the user and replaces the nan values with ‘ ‘.
   """
```

Step 3 : Import the Mapping File

```
def getmapping_Excel():
   """
   Gets the mapping excel from the user and replaces the nan values with ‘ ‘.
   """
```

Step 4 : Validation Trigger

```
def do_Validation():
   """
   Calls the validation function.
   """
```

Step 5 : Doing Validation

```
def Validate():
   """
   Does the actual validation and stores the validation file as a global variable.
It has several nested functions as listed below:
   """

  def compare_BankEndDate(df, r)://it is already done before for Mapping file right,
  why do it again for validation file?
```

```
    """
    :param df: The dataframe where the status of bank_end_dates need to be checked
    :param r: The no. of rows of the dataframe


   This function for df checks for the activeness  of the bank accounts.
   If any date in the   'BANK_END_DATE' column is before the current date, i.e now,
    then it marks it as FALSE. Any other date or a blank value is assigned as TRUE.

    """
```

```
def   drop_valid(df):
    """
    :param df: The processes dataframe where the status of bank_end_dates need to
be checked

    Drops the inactive bank records and the rows from the dataframe based on the
TRUE/FALSE values.

    """
```

```
    def find_columns(x):
        """


        Args:
            x: A predefined set of column values
        """
This function basically defines the structure and the column values needed for
validation ( did not understand )
```

```
def fill(df, target, source, column_list, mapping_file):
        """


        Args:
            df: The blank validation dataframe
            target: The target file
            source: The source file
            column_list: The predefined columns needed for Validation
            mapping_file: The mapping File
        """
The function calls several other functions to prepare the validation file.
The function initially gets the row indices for the particular gsl9 pair of source and
target based on Mapping file.
If the returned indices are of length one, then using those indices the validation file is
populated for all the columns.( Not clear [ on basis of previous unclear mapping
concepts] )
If the returned indices length is less than 1 ,then no match exists and the record is
skipped.
```

But if the returned indices are of length > 1 , that means multiple matching exists. The function findMatch is called.

def findMatch(rgsl,zgsl):// You had a similar function called getTargetIndex where you performed fuzzy logic..

```
""

    Args:
        rgsl: Target Mapping GSL
        zgsl: Source Mapping GSL

    """
```

It takes in the source and target gsl9 values , and finds the corresponding row indices in the source and target file. Once we get it, we match them based on certain predefined columns and then generate a pairing of the indices , which is then returned back to the fill function , and all of those pairs get an entry in the Validation file. (We only need the index for the GSL9 in src and target right? Why match them?) The generated tuples in the list are also sorted and flattened out.

```
def float_to_int(df):
    """

    Args:
        df: Changes the Pandas forced float values back to int values
    """
```

```
def check(z, r, m, df):
    """

    Args:
        z: The source column
        r: The target column
        m: The match column
        df: The Validation file
    """
```

You compare the source column with the target column and place either true/false/null in the match column.

Step 6 : Generating Summary

```
def create_Summary():
  """
  Gets the target excel from the user and replaces the nan values with ‘ ‘.
  """

  def makeSum(for_each_col, j, d, df):
    """
```

```
        Args:
          for_each_col: The column name
          j: The row number in the Summary file
          d: The Summary file
          df: The Validation file ( name it something else)
        """
Calculates the number of true ,false and blank values and populates the summary file.
```

Step 7 : Export The Validation File

```
def save_Validation():
    """
    Saves the validation file as excel on the user system.
    """
```

Step 8 : Export the Summary file

```
def save_Summary():
    """
        Saves the validation summary file as excel on the user system.

    """
```

Step 9 : Exit Application

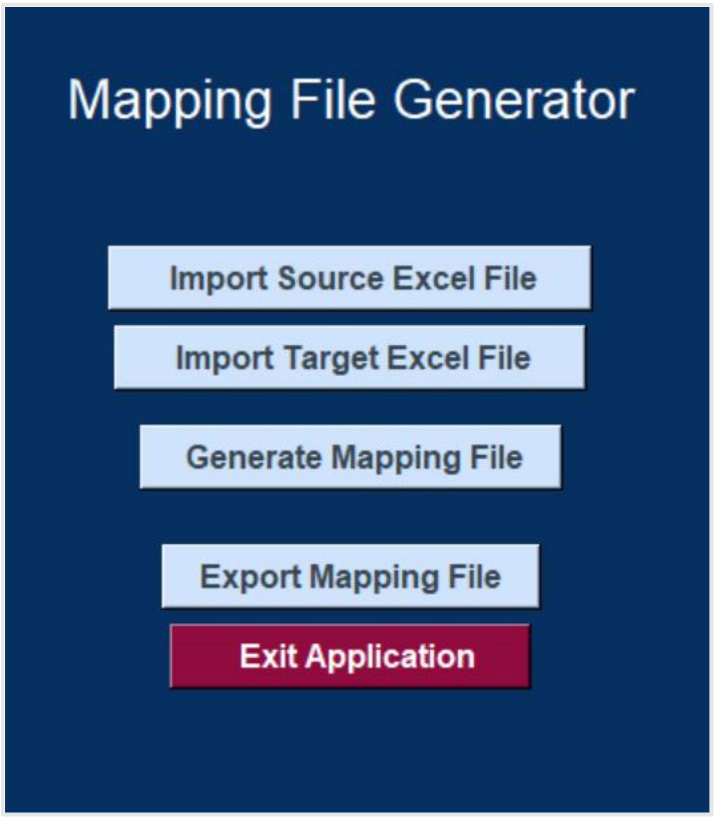
```
def exit():

Allows user to exit the application.    """
    """
```

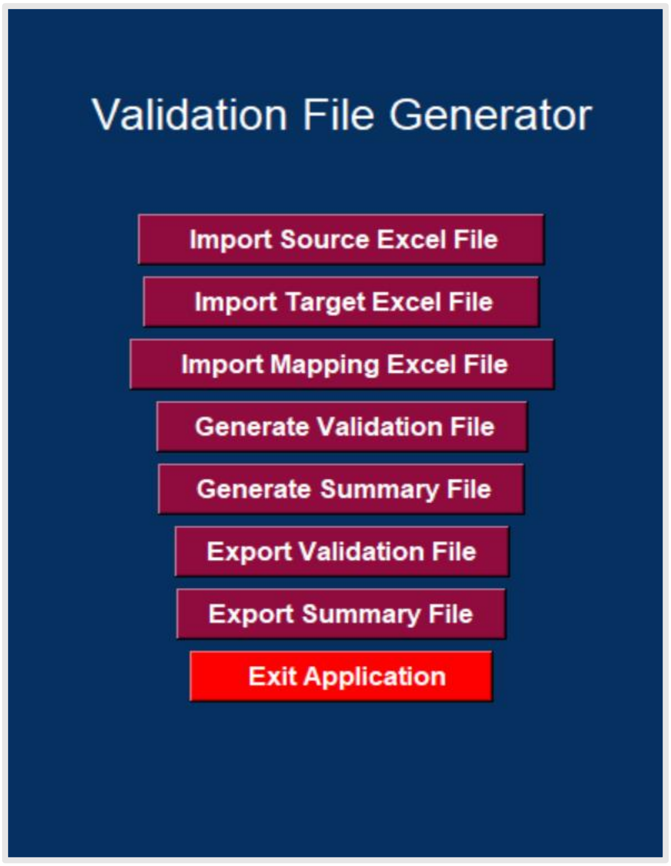
Since there are so many functions defined, the reader loses track as to which function performs which task and how it is connected to other functions. Please create a diagram in form of flow chart in the beginning, structuring the connections so as to depict the flow. Otherwise its difficult to comprehend where we are going to from which function. At the same time give a one line explanation for the function blocks you make in the diagram, so that when the user goes through it he has a helicopter level overview of entire validation aprt before he goes in deep to each function.

Testing

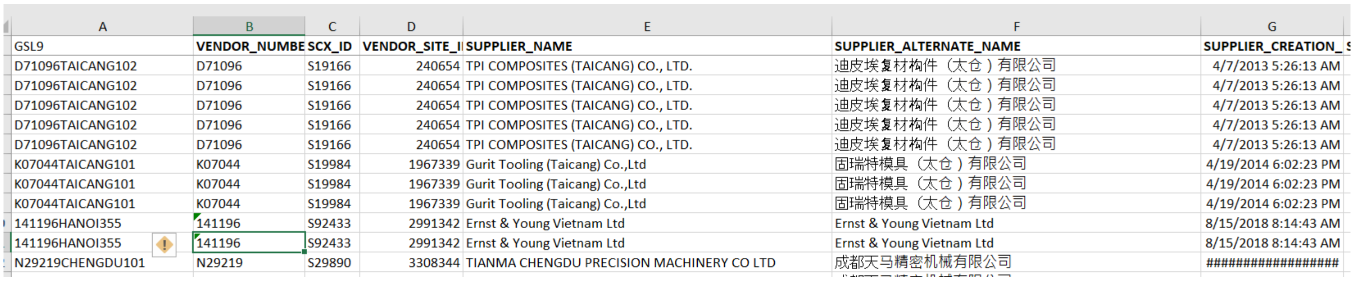
The model is tested on various input files in it’s latest update and has produced an accuracy of nearly 100%.

The GUI of both the application:


Mapping File Generator
Import Source Excel File
Import Target Excel File
Generate Mapping File
Export Mapping File
Exit Application

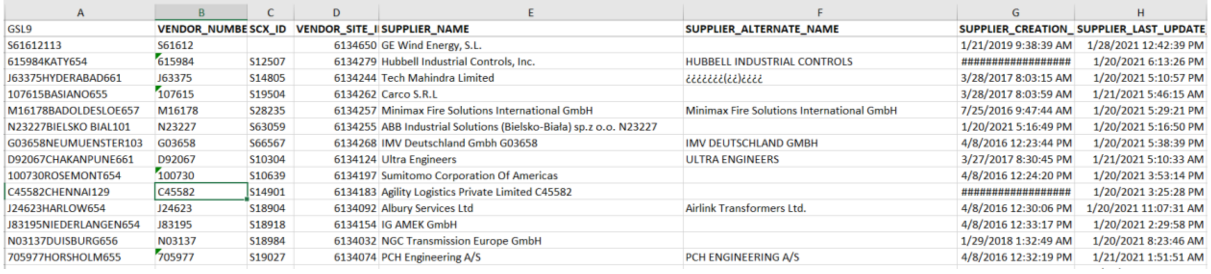

Validation File Generator
Import Source Excel File
Import Target Excel File
Import Mapping Excel File
Generate Validation File
Generate Summary File
Export Validation File
Export Summary File
Exit Application

Sample Testcase:



GSL9	VENDOR_NUMBE	SCX_ID	VENDOR_SITE_I	SUPPLIER_NAME	SUPPLIER_ALTERNATE_NAME	SUPPLIER_CREATION_
D71096TAICANG102	D71096	S19166	240654	TPI COMPOSITES (TAICANG) CO., LTD.	迪皮埃复材构件（太仓）有限公司	4/7/2013 5:26:13 AM
D71096TAICANG102	D71096	S19166	240654	TPI COMPOSITES (TAICANG) CO., LTD.	迪皮埃复材构件（太仓）有限公司	4/7/2013 5:26:13 AM
D71096TAICANG102	D71096	S19166	240654	TPI COMPOSITES (TAICANG) CO., LTD.	迪皮埃复材构件（太仓）有限公司	4/7/2013 5:26:13 AM
D71096TAICANG102	D71096	S19166	240654	TPI COMPOSITES (TAICANG) CO., LTD.	迪皮埃复材构件（太仓）有限公司	4/7/2013 5:26:13 AM
D71096TAICANG102	D71096	S19166	240654	TPI COMPOSITES (TAICANG) CO., LTD.	迪皮埃复材构件（太仓）有限公司	4/7/2013 5:26:13 AM
K07044TAICANG101	K07044	S19984	1967339	Gurit Tooling (Taicang) Co.,Ltd	固瑞特模具（太仓）有限公司	4/19/2014 6:02:23 PM
K07044TAICANG101	K07044	S19984	1967339	Gurit Tooling (Taicang) Co.,Ltd	固瑞特模具（太仓）有限公司	4/19/2014 6:02:23 PM
K07044TAICANG101	K07044	S19984	1967339	Gurit Tooling (Taicang) Co.,Ltd	固瑞特模具（太仓）有限公司	4/19/2014 6:02:23 PM
141196HANOI355	141196	S92433	2991342	Ernst & Young Vietnam Ltd	Ernst & Young Vietnam Ltd	8/15/2018 8:14:43 AM
141196HANOI355	141196	S92433	2991342	Ernst & Young Vietnam Ltd	Ernst & Young Vietnam Ltd	8/15/2018 8:14:43 AM
N29219CHENGDU101	N29219	S29890	3308344	TIANMA CHENGDU PRECISION MACHINERY CO LTD	成都天马精密机械有限公司	##################

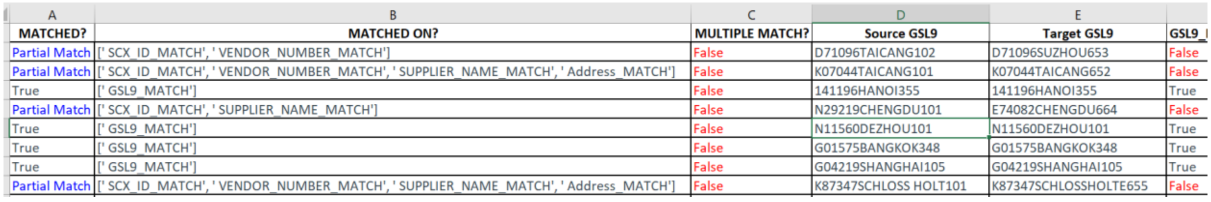
Zeal Shenyang Supplier Raw Data



GSL9	VENDOR_NUMBE	SCX_ID	VENDOR_SITE_I	SUPPLIER_NAME	SUPPLIER_ALTERNATE_NAME	SUPPLIER_CREATION_	SUPPLIER_LAST_UPDATE
S61612113	S61612		6134650	GE Wind Energy, S.L.		1/21/2019 9:38:39 AM	1/28/2021 12:42:39 PM
615984KATY654	615984	S12507	6134279	Hubbell Industrial Controls, Inc.	HUBBELL INDUSTRIAL CONTROLS	##################	1/20/2021 6:13:26 PM
J63375HYDERABAD661	J63375	S14805	6134244	Tech Mahindra Limited	¿¿¿¿¿¿¿(¿¿)¿¿¿¿	3/28/2017 8:03:15 AM	1/20/2021 5:10:57 PM
107615BASIANO655	107615	S19504	6134262	Carco S.R.L		3/28/2017 8:03:59 AM	1/21/2021 5:46:15 AM
M16178BADOLDESLOE657	M16178	S28235	6134257	Minimax Fire Solutions International GmbH	Minimax Fire Solutions International GmbH	7/25/2016 9:47:44 AM	1/20/2021 5:29:21 PM
N23227BIELSKO BIAL101	N23227	S63059	6134255	ABB Industrial Solutions (Bielsko-Biała) sp.z o.o. N23227		1/20/2021 5:16:49 PM	1/20/2021 5:16:50 PM
G03658NEUMUENSTER103	G03658	S66567	6134268	IMV Deutschland Gmbh G03658	IMV DEUTSCHLAND GMBH	4/8/2016 12:23:44 PM	1/20/2021 5:38:39 PM
D92067CHAKANPUNE661	D92067	S10304	6134124	Ultra Engineers	ULTRA ENGINEERS	3/27/2017 8:30:45 PM	1/21/2021 5:10:33 AM
100730ROSEMONT654	100730	S10639	6134197	Sumitomo Corporation Of Americas		4/8/2016 12:24:20 PM	1/20/2021 3:53:14 PM
C45582CHENNAI129	C45582	S14901	6134183	Agility Logistics Private Limited C45582		##################	1/20/2021 3:25:28 PM
J24623HARLOW654	J24623	S18904	6134092	Albury Services Ltd	Airlink Transformers Ltd.	4/8/2016 12:30:06 PM	1/20/2021 11:07:31 AM
J83195NIEDERLANGEN654	J83195	S18918	6134154	IG AMEK GmbH		4/8/2016 12:33:17 PM	1/20/2021 2:29:58 PM
N03137DUISBURG656	N03137	S18984	6134032	NGC Transmission Europe GmbH		1/29/2018 1:32:49 AM	1/20/2021 8:23:46 AM
705977HORSHOLM655	705977	S19027	6134074	PCH Engineering A/S	PCH ENGINEERING A/S	4/8/2016 12:32:19 PM	1/21/2021 1:51:51 AM

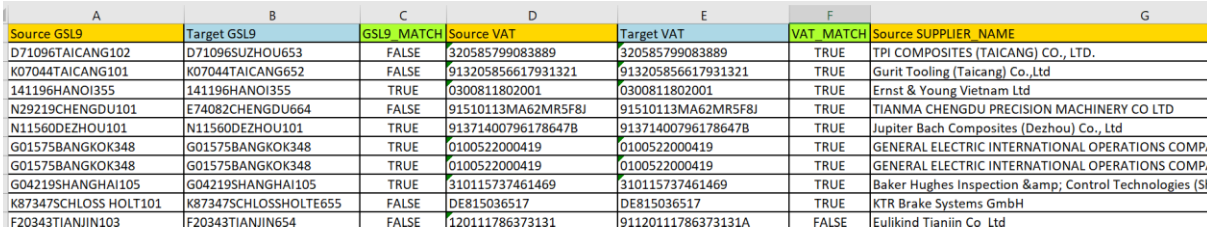
Races Shenyang UAT Load Supplier Data

Output of the Mapping File:



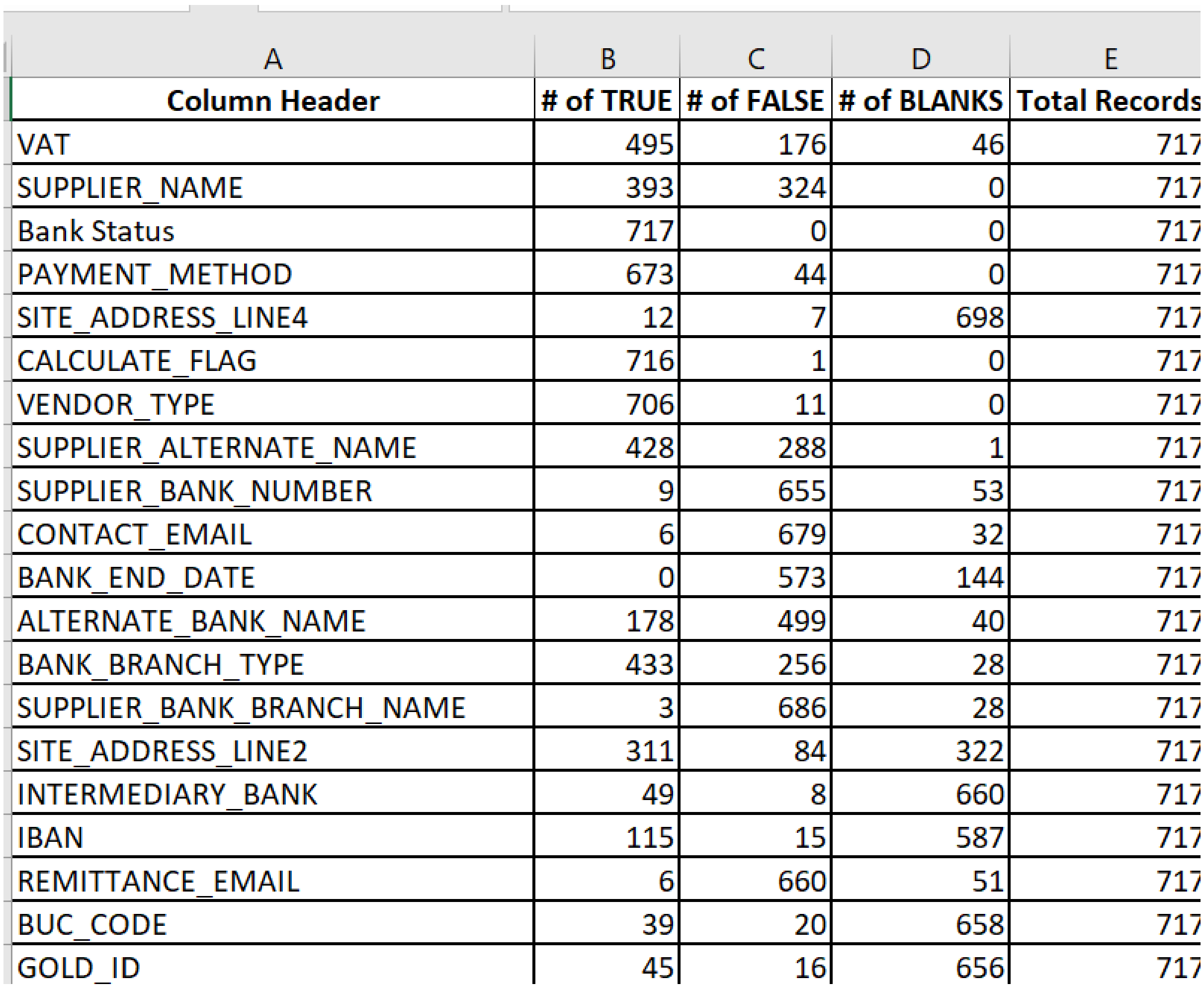
MATCHED?	MATCHED ON?	MULTIPLE MATCH?	Source GSL9	Target GSL9	GSL9_
Partial Match	[' SCX_ID_MATCH', ' VENDOR_NUMBER_MATCH']	False	D71096TAICANG102	D71096SUZHOU653	False
Partial Match	[' SCX_ID_MATCH', ' VENDOR_NUMBER_MATCH', ' SUPPLIER_NAME_MATCH', ' Address_MATCH']	False	K07044TAICANG101	K07044TAICANG652	False
True	[' GSL9_MATCH']	False	141196HANOI355	141196HANOI355	True
Partial Match	[' SCX_ID_MATCH', ' SUPPLIER_NAME_MATCH']	False	N29219CHENGDU101	E74082CHENGDU664	False
True	[' GSL9_MATCH']	False	N11560DEZHOU101	N11560DEZHOU101	True
True	[' GSL9_MATCH']	False	G01575BANGKOK348	G01575BANGKOK348	True
True	[' GSL9_MATCH']	False	G04219SHANGHAI105	G04219SHANGHAI105	True
Partial Match	[' SCX_ID_MATCH', ' VENDOR_NUMBER_MATCH', ' SUPPLIER_NAME_MATCH', ' Address_MATCH']	False	K87347SCHLOSS HOLT101	K87347SCHLOSSHOLTE655	False

Output of The Validation File:



Source GSL9	Target GSL9	GSL9_MATCH	Source VAT	Target VAT	VAT_MATCH	Source SUPPLIER_NAME
D71096TAICANG102	D71096SUZHOU653	FALSE	320585799083889	320585799083889	TRUE	TPI COMPOSITES (TAICANG) CO., LTD.
K07044TAICANG101	K07044TAICANG652	FALSE	913205856617931321	913205856617931321	TRUE	Gurit Tooling (Taicang) Co.,Ltd
141196HANOI355	141196HANOI355	TRUE	0300811802001	0300811802001	TRUE	Ernst & Young Vietnam Ltd
N29219CHENGDU101	E74082CHENGDU664	FALSE	91510113MA62MR5F8J	91510113MA62MR5F8J	TRUE	TIANMA CHENGDU PRECISION MACHINERY CO LTD
N11560DEZHOU101	N11560DEZHOU101	TRUE	91371400796178647B	91371400796178647B	TRUE	Jupiter Bach Composites (Dezhou) Co., Ltd
G01575BANGKOK348	G01575BANGKOK348	TRUE	0100522000419	0100522000419	TRUE	GENERAL ELECTRIC INTERNATIONAL OPERATIONS COMP/
G01575BANGKOK348	G01575BANGKOK348	TRUE	0100522000419	0100522000419	TRUE	GENERAL ELECTRIC INTERNATIONAL OPERATIONS COMP/
G04219SHANGHAI105	G04219SHANGHAI105	TRUE	310115737461469	310115737461469	TRUE	Baker Hughes Inspection &amp; Control Technologies (Sl
K87347SCHLOSS HOLT101	K87347SCHLOSSHOLTE655	FALSE	DE815036517	DE815036517	TRUE	KTR Brake Systems GmbH
F20343TIANJIN103	F20343TIANJIN654	FALSE	120111786373131	91120111786373131A	FALSE	Eulikind Tianjin Co Ltd

Output of the Summary File:



A	B	C	D	E
Column Header	**# of TRUE**	**# of FALSE**	**# of BLANKS**	**Total Records**
VAT	495	176	46	717
SUPPLIER_NAME	393	324	0	717
Bank Status	717	0	0	717
PAYMENT_METHOD	673	44	0	717
SITE_ADDRESS_LINE4	12	7	698	717
CALCULATE_FLAG	716	1	0	717
VENDOR_TYPE	706	11	0	717
SUPPLIER_ALTERNATE_NAME	428	288	1	717
SUPPLIER_BANK_NUMBER	9	655	53	717
CONTACT_EMAIL	6	679	32	717
BANK_END_DATE	0	573	144	717
ALTERNATE_BANK_NAME	178	499	40	717
BANK_BRANCH_TYPE	433	256	28	717
SUPPLIER_BANK_BRANCH_NAME	3	686	28	717
SITE_ADDRESS_LINE2	311	84	322	717
INTERMEDIARY_BANK	49	8	660	717
IBAN	115	15	587	717
REMITTANCE_EMAIL	6	660	51	717
BUC_CODE	39	20	658	717
GOLD_ID	45	16	656	717